Bases SQL

Interrogation: Syntaxe

```
Requête::= SELECT [DISTINCT|ALL] {ListeColonne |expression|*|(requête)[[AS]alias]}
                    FROM {  table  |  exp_join  | ( requête) }  [ alias ]
                                [, {  table  |  exp_join | ( requête) } [ alias
] ] ...
                    [WHERE condition]
                    [GROUP BY    exp [ , exp ] ... [HAVING condition]]
                     [ { UNION | UNION ALL | INTERSECT | MINUS }  requête ]
                    [ORDER BY {exp| position} [ASC |DESC]
[,{exp|position}[ASC|DESC]] ... ]
```

SELECT= construire une relation:

- Où exp_join est : { table | exp_join | (requête) } [alias]

  {INNER | LEFT | RIGHT} JOIN

  { table | exp_join | (requête) } [alias]

NOTATION:

{ }	Occurrence obligatoire
[ ]	Occurrence facultative
[ ] …	Répétition facultative
|	Alternative

Commande SELECT

Permet de :

- Sélectionner certaines lignes d'une table en fonction de leur contenu (sélection) ;
- Sélectionner certaines colonnes d'une table (projection) ;
- Combiner des informations venant de plusieurs tables (*jointure, union, intersection, différence, etc*) ;
- Combiner entre elles ces différentes opérations.

Une requête (*i.e.* une interrogation) est une combinaison d'opérations portant sur des tables (relations) et dont le résultat est lui-même une table dont l'existence est éphémère (le temps de la requête).

SELECT

```
SELECT [DISTINCT|ALL] {ListeColonne |expression|*|(requête)[[AS]alias]}
FROM <liste_de_tables>
```

SELECT est suivi de :

- noms de colonnes existantes dans les tables spécifiée après FROM et JOIN,
- expressions construites sur ces colonnes, via des opérateurs ou des fonctions

- ***DISTINCT:*** élimine les lignes identiques dans le résultat

- ***ALL:*** prend en compte les duplicatas (option par défaut)

- ***expression***: nom de colonne, fonction, constante ou calcul

- ***FROM:*** désigne la table à interroger

Sélection des lignes

❑ **Sélection de toutes les lignes et toutes les colonnes:**

Utilisation du symbole *: extrait toutes les colonnes de la table

```
SELECT * FROM nom_table
```

Exemple :

Pour connaître toutes les caractéristiques des pilotes:

```
SELECT * FROM Pilote ;
```

PILOTE

numP	nom	nbHVol	comp	villeP
1200	Gratien Viel	450	AF	Marseille
300	Didier Donsez	0	AF	Paris
4001	Richard Grin	1000	SING	Munich

Sélection des lignes

❑ **Sélection de toutes les lignes mais certaines colonnes:**

```
SELECT colonne, ...,colonne FROM nom_table
```

Exemple :

- Pour connaître les numéros des pilotes

```
SELECT numP FROM PILOTE;
```

numP
1200
300
4001

- Pour connaître les numéros et les compagnies des pilotes

```
SELECT numP, comp FROM
PILOTE;
```

numP	comp
1200	AF
300	AF
4001	SING

Sélection des lignes

❑ Suppression des lignes dupliquées:

```
SELECT DISTINCT colonne, ...,colonne
FROM   table ;
```

Exemple :

Pour connaître les compagnies pour lesquelles les pilotes ont travaillé:

Avec duplicata

```
SELECT comp FROM Pilote;
```

comp
AF
AF
SING

Sans duplicata

```
SELECT DISTINCT comp FROM Pilote;
```

comp
AF
SING

Colonnes calculées

```
SELECT      exp [AS alias], ...
FROM  table ;
```

- Une expression exp est : - une valeur

  - un nom de colonne

  - une fonction appliquée à une expression

  - une combinaison d'expressions à l'aide :

    - des opérations arithmétiques : +, -, *, /

    - de l'opérateur de concaténation : ||

- Un alias est un identificateur. Il est le nom de la nouvelle colonne dérivée par calcul de l'expression à laquelle il se rapporte.

Colonnes calculées

Exemple :

```
SELECT      numP, nbHVol, nbhvol*nbhvol AS auCarre, 10*nbhvol
FROM  Pilote;
```

numP	nbhvol	AUCARRE	10*nbhvol
1200	450	202500	4500
300	0	0	0
4001	1000	1000000	10000

Colonnes calculées

- **Sur les chaînes de caractères:**

Afficher les trajets assurés par les vols sous la forme : Ville de départ –– > Ville d'arrivée

```
SELECT numVOL,    VilleD||' --> '||VilleA  Trajet
FROM vol ;
```

NUMVOL	TRAJET
1	MARSEILLE --> PARIS
2	PARIS --> NY

- ||: l'opérateur de concaténation. Permet de concaténer des expressions (colonnes, calculs, fonctions ou constantes). La colonne résultante est considérée comme une chaîne de caractères.

Colonnes calculées

Exemple:

```
select numP, nom||' vole pour '|| comp AS "EMBAUCHE"
 from pilote;
```

PILOTE

numP	nom	nbhvol	comp	villeP
1200	Gratien Viel	450	AF	Marseille
300	Didier Donsez	0	AF	Paris
4001	Richard Grin	1000	SING	Munich

numP	EMBAUCHE
1200	Gratien Viel vole pour AF
300	Didier Donsez vole pour AF
4001	Richard Grin vole pour SING

En résumé

- Clause SELECT= projection (d'attributs)
- Clause FROM= liste des tables utilisées
- Renommage d'attribut (AS) et alias de table
- **Syntaxe de requête SQL** (crochets=option):

```
SELECT att1 [, att2 [AS att2'],…]
FROM nom_table1 [,nom_table2 [alias]…];
```

Restriction

```
SELECT [DISTINCT|ALL] {ListeColonne |expression|*|(requête)[[AS]alias]}
FROM {  table  |  exp_join  | ( requête) }  [ alias ]}
 WHERE condition
```

Avec Condition composée de colonnes, d'expressions, de constantes liées 2 à 2 entre des opérateurs:

- de comparaison

=	Égal
<>	différent
>	Supérieur
>=	Supérieur ou égal
<	Inférieur
<=	Inférieur ou égal

- logiques

NOT
AND
OR

-intégrés

BETWEEN
IN
LIKE
IS NULL

Recherche par comparaison

```
SELECT      ...
FROM  table
WHERE exp op_de_comparaison exp ;
```

Les opérateurs de comparaison sont:

=	Égal
<>	différent
>	Supérieur
>=	Supérieur ou égal
<	Inférieur
<=	Inférieur ou égal

Recherche par comparaison

Exemple

Pour connaître les noms des pilotes qui ont un nombre d'heures de vol supérieur à 200:

PILOTE

numP	nom	nbhvol	comp	villeP
1200	Gratien Viel	450	AF	Marseille
300	Didier Donsez	0	AF	Paris
4001	Richard Grin	1000	SING	Munich

```
SELECT      nom
FROM  PILOTE
WHERE nbhvol > 200 ;
```

Recherche par ressemblance

Opérateur **LIKE** (expression): compare de manière générique des chaînes de caractères à une expression.

```
SELECT        ...
FROM  table
WHERE exp_chaîne [NOT] LIKE motif [caractères spéciaux];
```

- Caractères spéciaux :
  - % : remplace 0 ou 1 ou plusieurs caractères
  - _ : remplace 1 caractère

Recherche par ressemblance

Exemples :

Pour connaître le nombre d'heures de vols de tous les Boeing :

```
SELECT      nbHVolA
FROM  Avion
WHERE AvNom LIKE 'Boeing%' ;
```

Avion

numAv	AvNom	nbHVolA	capAv	Loc	comp

Pour connaître tous les pilotes dont le nom contient la lettre "a" en deuxième position :

```
SELECT nom
FROM  Pilote
WHERE nom LIKE '_a%' ;
```

Recherche par condition conjonctive

```
SELECT      ...
FROM  table
WHERE condition AND condition ;
```

Exemple :

Pour connaître tous les avions qui sont à Marseille et dont la capacité est de 300 places :

```
SELECT      NumAv
FROM  Avion
WHERE Loc = 'Marseille'
AND   CapAv = 300 ;
```

Recherche par condition disjonctive

```
SELECT      ...
FROM  table
WHERE condition OR condition ;
```

Exemple :

Pour connaître tous les avions qui ont une capacité de 200 et 300 :

```
SELECT      numAv
FROM  Avion
WHERE capAv = 200 OR capAv = 300 ;
```

Recherche par condition négative

```
SELECT      ...
FROM  table
WHERE  NOT condition;
```

Exemple :

Pour connaître tous les avions qui ne sont pas localisés à Marseille

```
SELECT      numAv
FROM  Avion
WHERE NOT Loc = 'Marseille' ;
```

Recherche avec intervalle

Opérateur ***BETWEEN*** *limiteinf AND limiteSup:* teste l'appartenance à un intervalle de valeurs

```
SELECT       ...
FROM  table
WHERE  exp [NOT] BETWEEN exp AND exp;
```

Exemple :

Pour connaître tous les pilotes qui ont un nombre d'heures de vols entre 399 et 1000:

```
SELECT       numP
FROM  Pilote
WHERE nbhvol BETWEEN 399 AND 1000;
```

Recherche avec une liste

Les opérateurs permettant de comparer une valeur à un ensemble de valeurs sont :

- Opérateur ***IN*** *:* compare une expression avec une liste de valeurs
- Opérateur **ANY** *:* la comparaison est vraie si elle est vraie pour au moins un des éléments de l'ensemble.
- Opérateur **ALL:** la comparaison sera vraie si elle est vraie pour tous les éléments de l'ensemble

Recherche avec une liste

Opérateur ***IN (liste valeurs):*** compare une expression avec une liste de valeurs

```
SELECT        ...
FROM  table
WHERE  exp [NOT] IN (exp,….,exp);
```

Exemple :

Pour connaître tous les avions qui sont localisés soit à Marseille soit à Nice :

```
SELECT        NumAv
FROM   Avion
WHERE  Loc IN ('Marseille','Nice') ;
```

Recherche avec une liste

Opérateur **ANY** *:* la comparaison est vraie si elle est vraie pour au moins un des éléments de l'ensemble.

```
SELECT       ...
FROM  table
WHERE  exp op ANY (exp,…..,exp);
```

avec op ∈ { = , <> , > , >= , < , <= }

Opérateur **ALL:** la comparaison sera vraie si elle est vraie pour tous les éléments de l'ensemble

```
SELECT       ...
FROM  table
WHERE  exp op ALL(exp,…..,exp);
```

Recherche avec une liste

```
SELECT        numP
FROM  Pilote
WHERE nbHVol > ANY (100,399,1000);
```

Le nbhvol doit être supérieur à au moins une des valeurs

```
SELECT        numP
FROM  Pilote
WHERE nbhvol > ALL (100,399,1000);
```

Le NbHVol doit être supérieur au maximum de toutes les valeurs (donc plus que 1000)

```
exp = ANY (exp, ...,exp)
        ⇕
exp IN (exp, ...,exp)
```

```
exp <> ALL (exp, ...,exp)
        ⇕
exp NOT IN (exp, ...,exp)
```

Traitement de l'absence de valeurs

Rechercher l'absence de valeur

```
SELECT       ...
FROM  table
WHERE exp IS [NOT] NULL ;
```

Exemple :

Pour connaître tous les vols auxquels on n'a pas encore affecté d'avions :

```
SELECT       NumVol
FROM  Vol    WHERE NumAv IS NULL ;
```

```
   NUMVOL
---------
      500
```

NUMVOL	NUMP	NUMAV	DATEA	DATED	VILLEA	VILLED
1	300	200	12/02/18	11/02/18	PARIS	MARSEILLE
2	4001	300	13/05/18	12/05/18	NW	Paris
490	7006	800	19/09/18	(null)	Paris	NW
500	(null)	(null)	14/09/18	13/09/18	Nice	Marseille

Traitement de l'absence de valeurs

Donner une valeur à l'absence de valeur

- fonction NVL remplace une valeur nulle avec une valeur non nulle

NVL(exp1,exp2) = exp1 si elle est définie

= exp2 sinon

- *NVL(exp1, exp2):* retourne exp2 si exp1 est nulle, sinon la fonction retourne exp1.
- exp1 et exp2 doivent être de même type.

Traitement de l'absence de valeurs

Exemple :

Pour faire en sorte qu'une capacité d'avion inconnue soit considérée comme une capacité nulle.

```
SELECT      NumAv, nvl(CapAv,0) CapAv
FROM  Avion ;
```

NUMAV	AVNOM	NBHVOL	CAPAV	LOC	COMP
200	Boeing	30	100	Marseille	AF
300	Heinkel	250	2	Allemagne	SING
500	Sportavia	20	1	CHINE	SING
600	test	2		marseille	NW
800	boeing	4		marseille	NW

NUMAV	CAPAV
200	100
300	2
500	1
600	0
800	0

Ordonner les réponses

ORDER BY :permet de trier les lignes dans un résultat d'une requête SQL.

Syntaxe:

```
SELECT        ...
FROM  table
[WHERE        ... ]
ORDER BY      {exp|position}[ASC|DESC]
      [{exp|position}[ASC|DESC]]...;
```

- Par défaut les résultats sont classés par ordre ascendant (ASC)

- DESC: pour un tri par ordre descendant

Ordonner les réponses

Exemple :

Pour connaître les numéros et les noms des pilotes triés par ordre croissant des noms:

```
SELECT        numP,
nom
FROM   PILOTE
ORDER BY      nom ;
```

NUMP	NOM	NBHVOL	COMP	VILLEP
1200	Gratien Viel	450	AF	Paris
300	Didier Donsez	0	AF	Marseille
4001	Richard Grin	1000	SING	Monaco
7006	Fresnais	2450	SING	Nice
200	martin	400	AC	(null)
500	JACQUES	200	SING	Marseille
400	(null)	20	SING	(null)

```
     NUMP NOM
--------- --------------
      300 Didier Donsez
     7006 Fresnais
     1200 Gratien Viel
      500 JACQUES
      200 martin
     4001 Richard Grin
      400
```

Ordonner les réponses

Exemple :

Pour connaître les numéros de vols, et les dates d'arrivée triées par ordre croissant :

```
SELECT      NumVol, DateA
FROM  Vol
ORDER BY     NumVol, DateA ;
```

Exemple :

Pour connaître les avions par ordre alphabétique et par capacité **décroissante** :

```
SELECT      DISTINCT AvNom, CapAv
FROM  Avion
ORDER BY     AvNom, CapAv DESC ;
```

Ordonner les réponses

Exemple :

PILOTE

numP	nom	villeP
11	Martin	Marseille
14	Lucie	Rennes
12	Jean	Marseille
13	Marc	Lyon
2	Martin	Aix

```
SELECT      *
FROM  PILOTE
Order by villeP desc;
```

numP	nom	villeP
14	Lucie	Rennes
11	Martin	Marseille
12	Jean	Marseille
13	Marc	Lyon
2	Martin	Aix

```
SELECT      *
FROM  PILOTE
Order by villeP desc, nom;
```

numP	nom	villeP
14	Lucie	Rennes
12	Jean	Marseille
11	Martin	Marseille
13	Marc	Lyon
2	Martin	Aix

En résumé

- Clause WHERE= sélection (d'instances)
- Condition = combinaison (AND, OR) d'expressions (=, !=, <, <=,>,>=)
- Opérateurs spécifiques (IN, BETWEEN), valeur non définie (NULL)
- Clause ORDER By= tri du résultat
- Syntaxe de requête SQL (crochets=option):

```
SELECT att1 [, att2 [AS att2'],…]
FROM nom_table1 [,nom_table2 [alias],…]
[WHERE condition]
[ORDER BY atti [,attj,…]];
```